Announcements

Teaching Evaluation
- Need a volunteer!
- Different office hours for next week
  - Yana: M, W 4-5pm
  - Yana: appointment via email for Thu 8-10am
  - Surajit: Tu 1.00-2.00pm
  - Surajit: Thu 11.30am-12.30pm Office: 226D
- Continue to check mail archive for all communications/issues related to the course
- Final Exam Thu March 15
  - In Class (Loew 102) 2.30-4.20pm
  - 100 points, 100 mins
  - Questions – 5 to 10 points each

Selective Exclusion from Finals

- All concepts included.
- Exclusions only apply to "direct questions"
  - No direct questions on E-R Diagrams (Chapter 2, 3.1-3.4)
  - But, concepts of relationships, key, constraints included
  - No direct questions on deadlocks (Chapter 10.3 – Vol 2)
  - No direct questions on Media Failure (Chapter 8.5 – Vol 2)
  - No direct questions from "Undo-only" and "Redo-only" logging (Chapter 8.3-8.4 Vol 2) but there will be questions from "Undo/Redo Logging" (Chapter 8.5 Vol 2)
  - Be familiar with concepts in Chapters 8.3-8.4 (Vol 2)

Query Optimization

Required Reading: 7.2, 7.4, 7.5, 7.6, 7.7.0 – 7.7.2, 7.7.6

Query Optimization: Phases

- Parsing phase
  - Produces a parse tree
- Query Rewrite phase
  - Produces a logical tree
- Physical Query plan generation
  - Produces executable (physical) plan
Query Optimization

- Algebraic laws provide alternative execution plans
- Estimate costs of alternative modes of execution
- Efficiently search the space of alternatives
  - Simplify search by applying heuristics (without costing)
  - Apply laws that *seem* to result in cheaper plans

Converting from SQL to Logical Plans

Select a1, ..., an
From R1, ..., Rk
Where C

\[ \Pi_{a_1, \ldots, a_n}(\sigma_C(R_1 \bowtie R_2 \bowtie \ldots \bowtie R_k)) \]

Enumerating Physical Plans

- Exhaustive – Consider all possible:
  - Join Orders
  - Algorithms for each operator
- Heuristic Search
  - E.g. Greedy approach
  - Pick next relation such that join size is smallest

Enumerating Physical Plans

- Branch-and-Bound Enumeration
  - Find a good starting plan (having cost C)
  - In subsequent search, eliminate any subquery with cost > C
- Hill Climbing
  - Start with heuristically selected plan
  - Explore plans in the "neighborhood"
    - E.g. replace Nested-Loops join with Hash-Join

Determining Join Order

- Select-project-join
- Push selections down, pull projections up
- We need to choose the join order
- This is the main focus of our study today

- Dynamic Programming
  - Bottom-up strategy
  - For each subexpression, only keep plan with the least cost
    - Consider possible implementations of each node assuming
    - Must consider *interesting orders*
      - E.g., when subexpression is sorted on a sort attribute at the node
      - More later
Determining Join Order: Join Trees

For each subquery Q in \{R_1, ..., R_n\} compute the following:
1. \text{Size}(Q)
2. A best plan for Q: Plan(Q)
3. The cost of that plan: Cost(Q)

Linear Join Trees

Join Ordering Problem

Bushy Join Trees

Dynamic Programming (1)

Intro to Enumeration using Dynamic Programming

Dynamic Programming (1)

Step 1: For each \{R_i\} do:
1. \text{Size}((R_i)) = B(R_i)
2. Plan((R_i)) = R_i
3. \text{Cost}((R_i)) = (cost of scanning R_i)
Dynamic Programming (2)

- Step 1: For each $Q \subset \{R_1, \ldots, R_n\}$ of cardinality $i$ do:
  - Compute Size($Q$)
  - For every subquery $Q'$ and a relation $R_j$
    - $Q = Q' \cup \{R_j\}$
    - Compute cost(Plan($Q'$), Plan($R_j$))
    - Cost($Q$) = the smallest such cost
    - Plan($Q$) = the corresponding plan
- Final Step: Return Plan($\{R_1, \ldots, R_n\}$)

Comments on Enumeration

- Recall: computes optimal plans for subqueries:
  - Step 1: $\{R_1\}$, $\{R_2\}$, ..., $\{R_n\}$
  - Step 2: $\{R_1, R_2\}$, $\{R_1, R_3\}$, ..., $\{R_{n-1}, R_n\}$
  - ...
  - Step n: $\{R_1, \ldots, R_n\}$
- Read Example 7.3.5 (important)
- Practical Issues
  - Heuristics for Reducing the Search Space
  - Restrict to trees “without cartesian product”
  - Need more than just one plan for each subquery:
    - “Interesting orders”

Role of Interesting Order

- Join conditions: $R_1.a = R_3.a = R_5.a$
- Sub-optimal plan for first join need to be considered

Completing the Physical Query Plan

- Choose algorithm to implement each operator
  - Need to account for more than cost:
    - How much memory do we have?
    - Are the input operand(s) sorted?
  - Decide for each intermediate result (not included in exam):
    - To materialize or to pipeline

Choice of Algorithms: Selection

- Sequential scan
- Index-based
  - Identify selection condition for which an index exists
  - Retrieve all tuples using the index
  - How many are there?
  - How much does it cost?
  - Equi-join condition — easy
  - Inequality condition:
    - Clustering Index $B(R1)/R2$, Non-clustered Index $T(R1)/R2$
  - Post-Filter other predicates
- Read Example 7.37

Choice of Algorithms: Join

- One pass algorithms works for “small” sizes
- Sort-Join
  - When an order exists at least partially
  - Multiple joins on same attribute (interesting order)
- Index Nested Loop Join
  - Index on inner
- Hash Join
Physical Query Plan

- Leaves
  - Tablescan @, Indexscan(R,C), Indexscan (R,A)
- Selection
  - Filter @, Indexscan(R,C)
- Join
  - Hash-Join(),...
  - Sort()....

What did you learn?

- Database schema, its design principles
- SQL – Writing Database Programs
- Connectivity/Client Server Issues
- Inside DBMS
  - Transactions, Storage, Query Engine
- We did not cover:
  - Data Analysis, Warehouse, Mining
  - ETL Monitor (Application Server)
  - XML
  - Multimedia